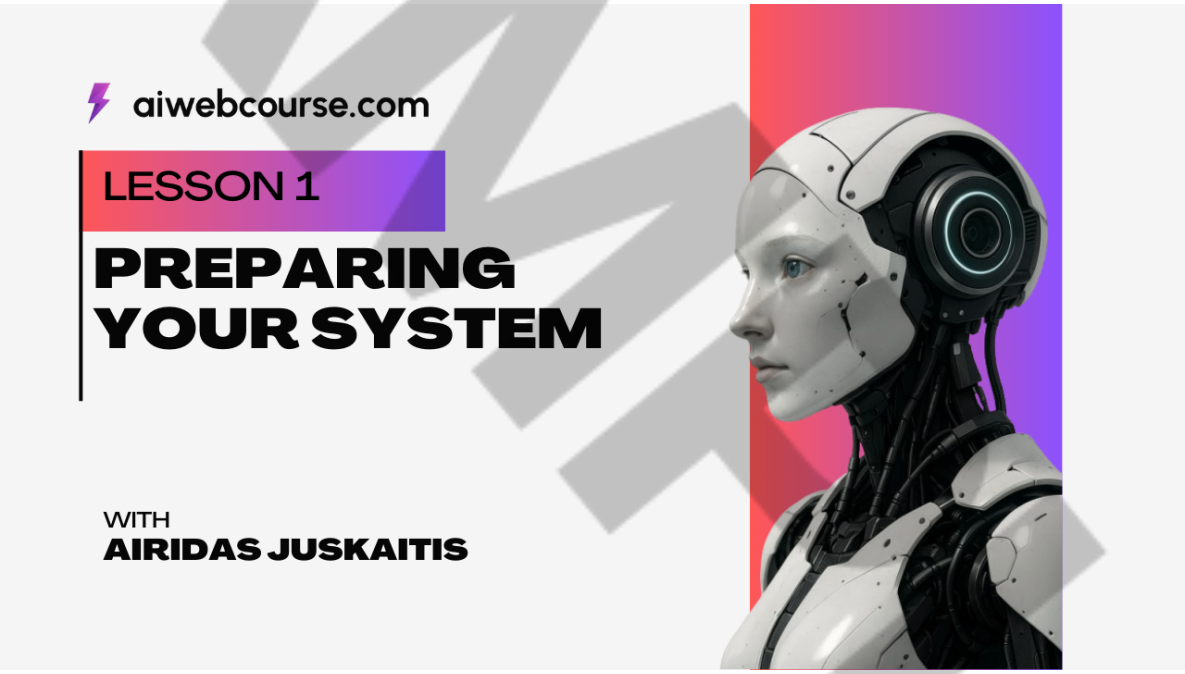
aiwebcourse.com

AI Web Course: Build Your AI Startup in 12 Hours Using CLINE

Lesson 1: Preparing Your System

Get your development environment ready with Visual Studio Code, CLINE AI assistant, and essential tools - no coding experience required.



What you'll learn

- Installing Visual Studio Code
- Installing CLINE VSCode plugin
- Getting CLINE to install PostgreSQL and Node.js (Mac only)
- Manual installation for Windows users
- Setting up Google Cloud account
- Verifying CLINE functionality with first tasks

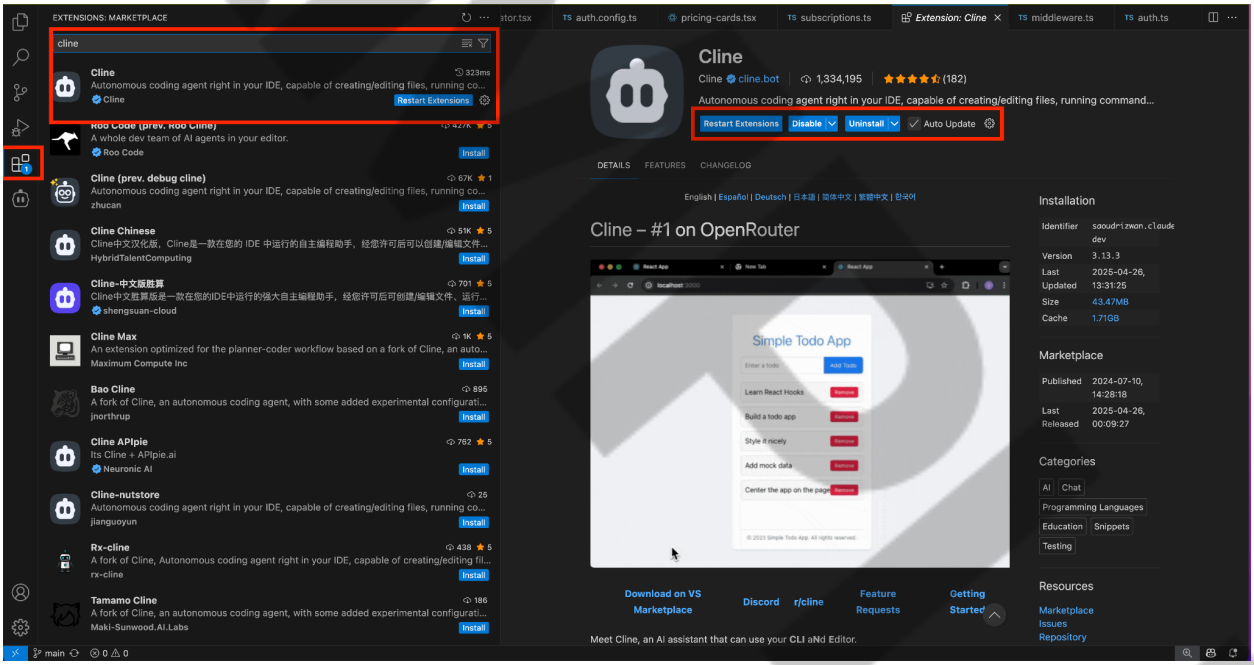
Step-by-Step Instructions:

1. Install Visual Studio Code

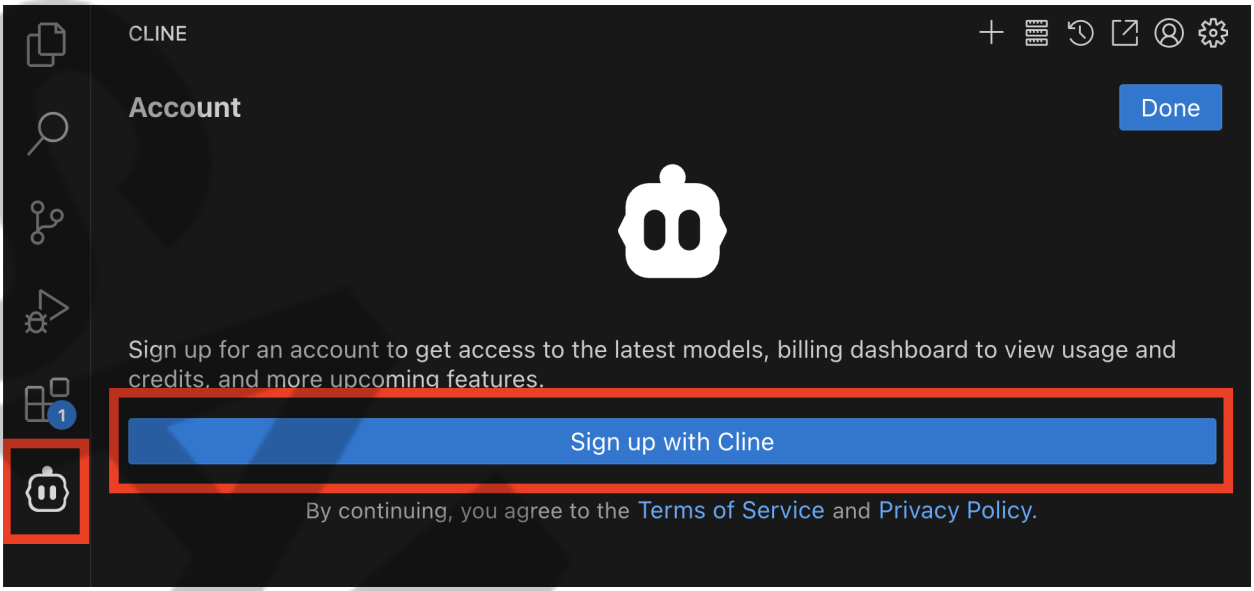
- Download from https://code.visualstudio.com/
- Follow the installation wizard for your operating system (Mac/Windows/Linux)

2. Install the CLINE VSCode Plugin

- Open Visual Studio Code
- Go to Extensions (icon on the sidebar)
- Search for "CLINE"
- Click Install



- Click the CLINE icon in the sidebar
- Click "Sign up with CLINE" and complete registration

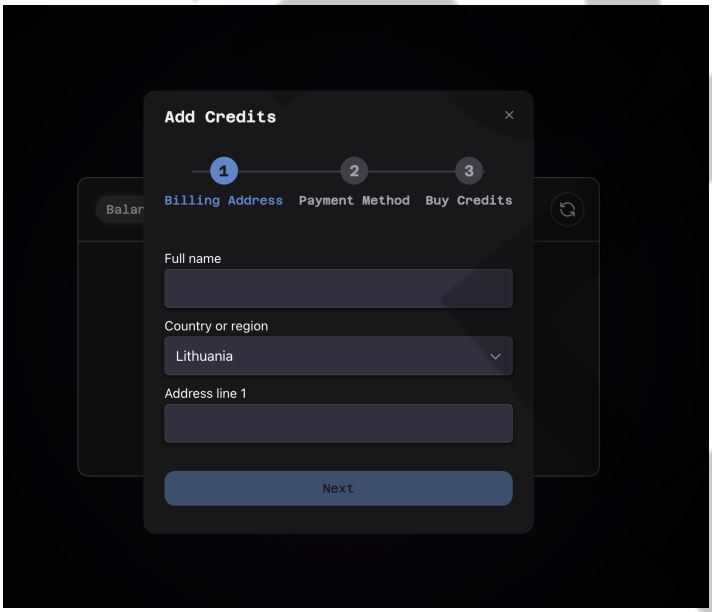


Once signed up, add credits to your CLINE account.

These credits are required to access the Claude Sonnet 3.7 model — the model we will use throughout the course because it consistently produces the most accurate results in CLINE. Credits are consumed **on demand** depending on the complexity and length of each task.

You can start with as little as **$10**, but we recommend **$50** to comfortably complete all lessons. For reference, getting to the end of Lesson 3 usually costs around **$20 worth of credits**.

When you click **Add Credits** in the CLINE interface, you'll be shown the billing window:



Complete the steps to purchase credits and unlock full access to CLINE's features.

> *Note: CLINE also lets you switch between different AI models, but in this course we will use **Claude Sonnet 3.7** exclusively.*

3. Install PostgreSQL and Node.js

To run and develop applications in this course, your computer needs two essential programs: **PostgreSQL** (a database server) and **Node.js** (a JavaScript runtime environment). These are both standard tools used in web development. While installing them might seem technical, they are simply required apps that power the code we build together.

For Mac Users:

- Open the CLINE terminal inside VSCode
- Prompt CLINE:

```
Please install PostgreSQL and Node.js on my machine.
```

- Follow any additional instructions CLINE provides to complete the setup

For Windows Users:

CLINE installation commands are not fully reliable on Windows. Please install manually:

PostgreSQL Installation:

- Go to https://www.postgresql.org/download/windows/
- Download the PostgreSQL installer for Windows
- pRun the installer and follow the default settings
- Restart your computer.
- Check if installation was successful by running cmd.exe (Command Prompt) and typing `pg_isready`.

Node.js Installation:

- Go to https://nodejs.org/en/download/
- Download the LTS (Long-Term Support) version for Windows
- Run the installer and complete the installation

After installation, verify everything works by opening a new terminal:

- **Mac users:** Open the "Terminal" application

To see full course, buy it at aiwebcourse.com

(Windows users: already manually installed.)

Step 4: Build a Simple TODO Application

Prompt CLINE:

```
Build me a simple TODO application in REACT.
```

Step 5: Customize the Application

Prompt CLINE:

```
Make the website look futuristic, I want indigo and purple colors, apply a
gradient background and make a logo saying aiwebcourse first app!
```

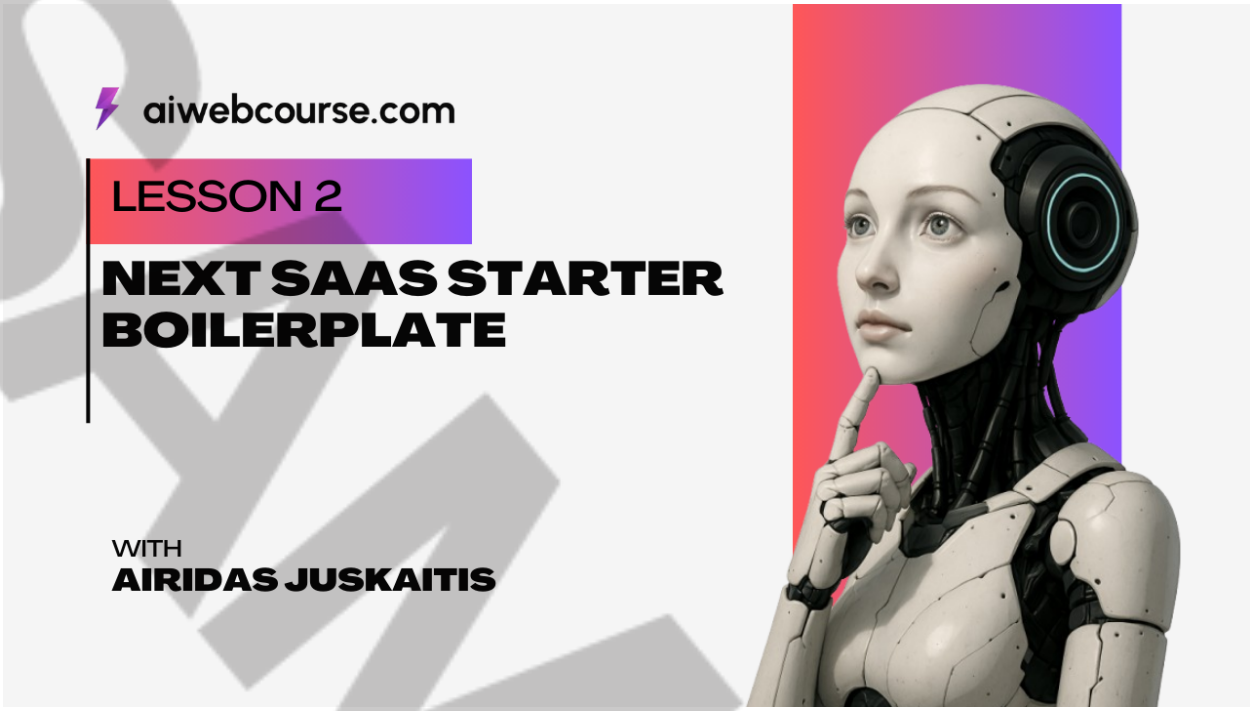
This will complete your first functional AI-built React application!

(You should have an example TODO app at this point)

You're ready to start building your SaaS!

Lesson 2: NEXT SAAS Starter Boilerplate

We'll now set up the core of our SaaS using a powerful open-source starter template built for Next.js 14.



Lesson 2 Plan:

- Downloading NEXT SAAS Starter boilerplate
- Setting up environmental variables:
    - Google Auth
    - Github Auth
    - Postgres Database
- Resend emailing service
- Stripe account and env variables
    - Creating a stripe subscription product
    - Creating a stripe web-hook
    - Local stripe web-hook setup for testing

Section 1: Introduction

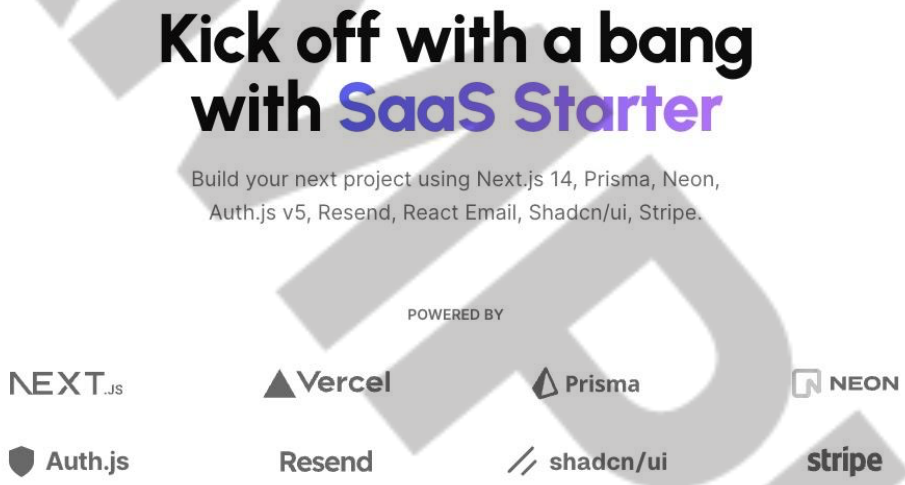
This project uses the free and powerful **Next SaaS Starter** template, available on GitHub:

GitHub Repo: https://github.com/mickasmt/next-saas-stripe-starter

This starter includes:

- Preconfigured Next.js 14 setup
- Built-in user authentication via Auth.js
- PostgreSQL database integration using Prisma
- Working Stripe payments flow
- Email integration with Resend
- Styled components via shadcn/ui

We will be using the open-source repository {pay to get link to repo} as the foundation of our application. It includes many of the SaaS essentials like authentication, billing, multi-tenancy, and more.



Section 2: Download and Launch the Boilerplate

1. Download the ZIP folder from the GitHub repo
2. Extract it locally
3. Open the folder in Visual Studio Code
4. Open the integrated terminal:

To see full course, buy it at aiwebcourse.com

To see full course, buy it at aiwebcourse.com

To see full course, buy it at aiwebcourse.com

🔐 Google Authentication

We recommend using or creating a **separate Gmail account** that you'll dedicate to this project.

Later in the course, this same account will be used to authenticate with Resend, Stripe, and Google Cloud.

1. Go to https://console.cloud.google.com and create a Google Cloud project
2. Claim your **$300 in free credits**, which are valid for 12 months
3. Navigate to **APIs & Services > Credentials**
4. Click **"Create OAuth client ID"**
5. Set the authorized **redirect URI** to:

```
http://localhost:3000/api/auth/callback/google
```

6. Set the **App Domain** to:

```
localhost:3000
```

7. Save and copy the **Client ID** and **Client Secret** into your `.env` file.

See screenshot below 👇

To see full course, buy it at aiwebcourse.com

Section 6: Setting Up Stripe Integration (Test Mode)

Stripe is a popular online payment processing platform used by companies of all sizes. It supports subscriptions, one-time payments, invoicing, and more. In our project, we use Stripe to manage the billing of our SaaS product. The starter template already includes a working integration with Stripe, which we'll now configure.

At this stage, we will set up **Stripe in test mode**, meaning all transactions are simulated and no real money is involved. This lets us safely test our billing flows.

🎯 Why Stripe?

- Handles recurring billing and subscriptions
- Integrates easily with modern stacks like Next.js
- Secure and trusted by developers
- Offers test mode with powerful debugging tools

At this point, we'll configure the Stripe-related `.env` variables:

```
STRIPE_API_KEY=
STRIPE_WEBHOOK_SECRET=

NEXT_PUBLIC_STRIPE_PRO_MONTHLY_PLAN_ID=
NEXT_PUBLIC_STRIPE_PRO_YEARLY_PLAN_ID=

NEXT_PUBLIC_STRIPE_BUSINESS_MONTHLY_PLAN_ID=
NEXT_PUBLIC_STRIPE_BUSINESS_YEARLY_PLAN_ID=
```

🧾 Create a Stripe Account

1. Go to https://dashboard.stripe.com/register and create a free Stripe account
2. After signing up, switch to **Test Mode** using the toggle in the top-left corner of the dashboard
3. Click on **Developers > API Keys** to get your `STRIPE_API_KEY`
4. Paste that into your `.env`

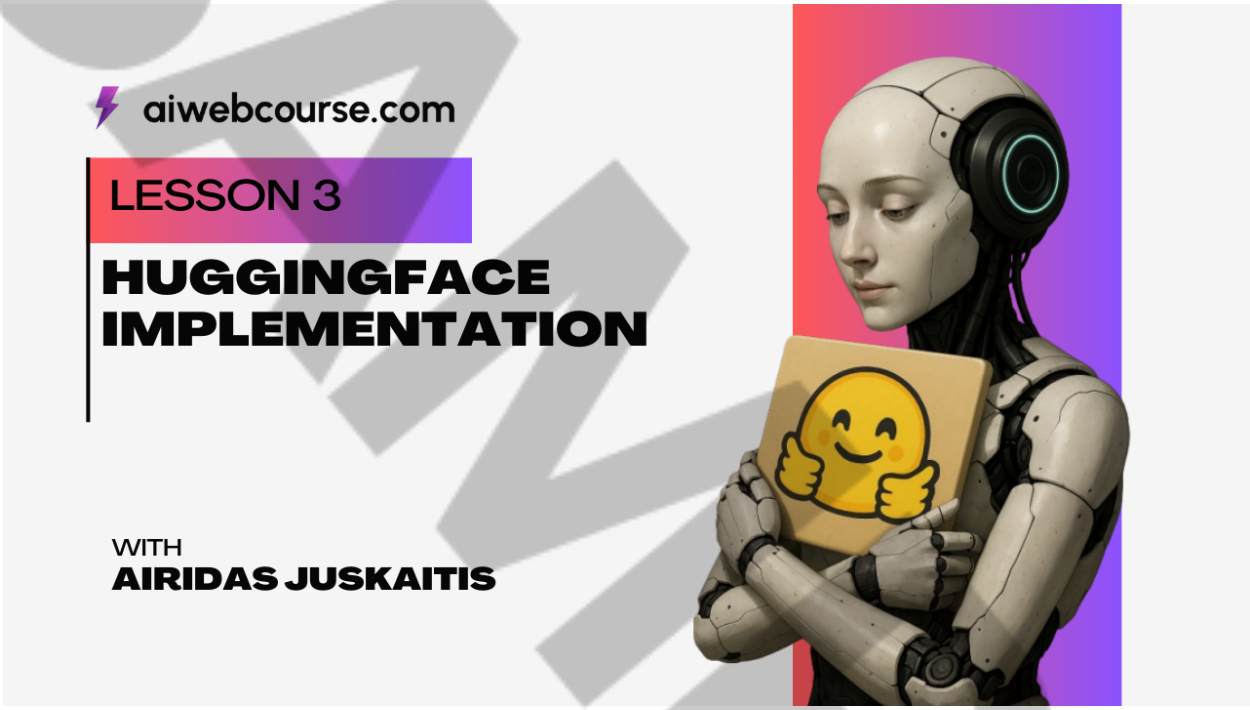
💳 Create Your Pricing Plans

1. In the Stripe dashboard, go to **Products**
2. Create two products:
    - **Pro Plan**

To see full course, buy it at aiwebcourse.com

Lesson 3: Hugging Face Integration

Build AI Features into Your SaaS Using CLINE + Hugging Face
Estimated time: ~1.5 hours



🧠 How This Lesson Works

The Hugging Face platform has a section called **SPACES** — this is where developers publish working demos of AI models using a framework called **Gradio**.

These Spaces are not just demos — they're **fully functional AI apps** that expose simple APIs. In this course, you'll learn how to:

- Find a Hugging Face Space that fits your product idea
- Understand its inputs/outputs
- Use CLINE to build front-end and back-end code to integrate that Space into your app

⚠️ **For this section you will have to create a Google Cloud bucket and make it public!**

To see full course, buy it at aiwebcourse.com

To see full course, buy it at aiwebcourse.com

To see full course, buy it at aiwebcourse.com

To see full course, buy it at aiwebcourse.com

To see full course, buy it at aiwebcourse.com

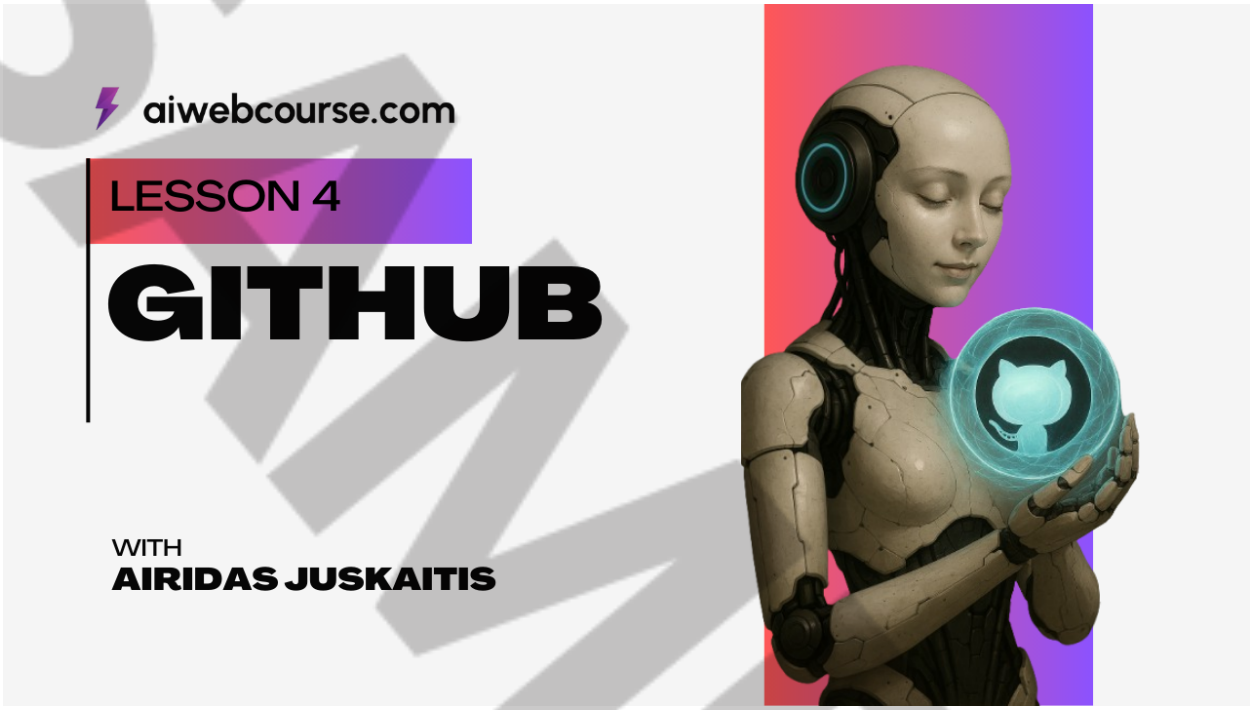
To see full course, buy it at aiwebcourse.com

To see full course, buy it at aiwebcourse.com

To see full course, buy it at aiwebcourse.com

Lesson 4: GitHub Version Control

Track Your Work, Collaborate with Confidence, and Save Your Startup's Code
Estimated time: ~1 hour



🔷 Lesson Overview

In this lesson, you'll learn how to connect your SaaS project to **GitHub** — the most widely used platform for collaborative coding and version control.

Whether you're working solo or as part of a team, GitHub helps you:

- Save code history
- Roll back changes
- Work on features safely using branches
- Collaborate with others in real-time

You'll use **Git**, the underlying technology behind GitHub, to track, save, and sync your project files.

🧩 What You'll Learn

- What Git and GitHub are, and why they matter
- Connecting your existing project to GitHub
- Git basics: `add`, `commit`, `push`, `pull`
- What the `.gitignore` file does and why it's critical
- How to collaborate using branches
- Viewing and tracking your project's history

🔧 Prerequisites

✅ GitHub account (created in Lesson 2)

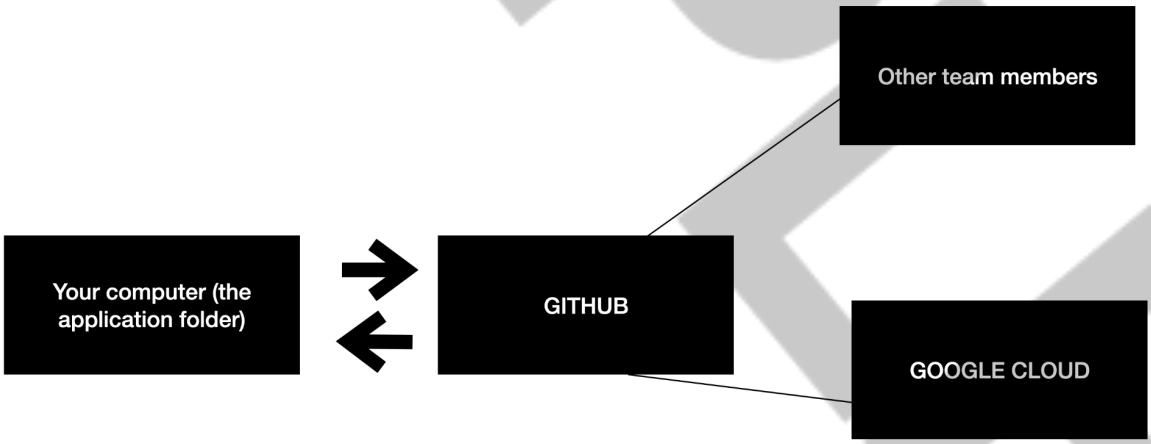
✅ Your SaaS codebase already working locally (from Lessons 1–3)

🔷 Section 1: What is GitHub?

GitHub is an online platform for hosting and collaborating on code using **Git** — a version control system that tracks every change you make.

Every time you save (commit) your code with Git, it creates a snapshot of your project. You can go back to earlier versions, share your code, or merge changes from collaborators.

Think of GitHub as "Google Drive for developers" — but with powerful tools for teamwork, code reviews, and history tracking.



We will use Github in this course to transfer our project to Google Cloud for deployment! 🙂

🔷 Section 2: Initialize Git in Your Project

1. Open the integrated terminal in your Visual Studio Code project
2. Run the following commands:

```
git init
git add .
git commit -m "Initial commit"
```

3. Create a new GitHub repository:
    - Go to https://github.com
    - Click “New”
    - Name it `aiwebcourse (or your preferred name)`
    - Leave it empty (don’t initialize with README)

4. Connect your local repo to GitHub:

```
git remote add origin https://github.com/yourusername/aiwebcourse.git
git branch -M main
git push -u origin main
```

💡 Now your entire SaaS project is backed up and trackable via GitHub.

🔷 Section 3: Git Fundamentals

Git Command	What It Does
`git add .`	*Stages all changes for commit*
`git commit -m "msg"`	*Saves a snapshot with a message*
`git push`	*Uploads local commits to GitHub*
`git pull`	*Syncs latest changes from GitHub*

`git status` *Shows what's changed or staged*

💡 You can use CLINE to perform these commands for you!

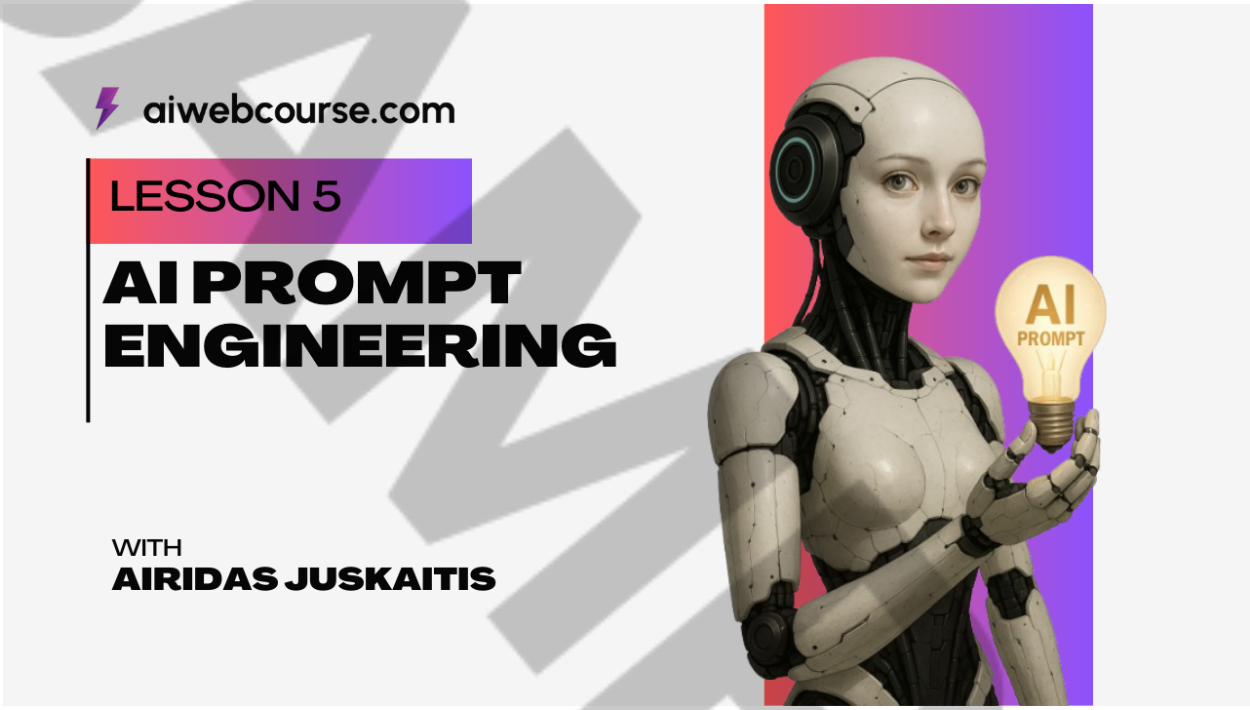
```
Please help me setup github, I want to save my project in github and I already have an account.
```

To see full course, buy it at aiwebcourse.com

To see full course, buy it at aiwebcourse.com

Lesson 5: AI Prompt Engineering for Web Application Enhancement

Track Your Prompts, Refine Your App, and Launch with Confidence
🕒 **Estimated time**: 3–4 hours



Lesson Overview

In this lesson, you'll harness the power of prompt engineering to enhance your web application's design, functionality, and user engagement. Using tools like **CLINE.bot**, and **Google Vertex AI Studio**, you'll apply specific prompts to improve visual appeal, UX, rate limiting, analytics integration, and final launch readiness.

By the end, your Ghibli-style image generation app will feature a compelling hero section, engaging blog content, working Stripe paywall logic, analytics tracking, and polished branding including a custom AI-generated logo.

What You'll Learn

- How to structure effective AI prompts to control output
- How to enhance your homepage's design with Tailwind components
- How to generate visuals like logos using **Google Cloud Vertex AI Studio**
- How to use prompts to add blogs, update UI, and manage content
- How to integrate analytics using **Google Analytics** and **Facebook Pixel**
- How to implement rate limiting and dynamic pricing logic with Stripe

---

Prerequisites

- Your SaaS app built in Lessons 1–4
- A GitHub project with working deployment
- Access to CLINE.bot
- Access to **Google Cloud Console** with **Vertex AI enabled**
- Stripe account (with webhooks set up) (from Lesson 2)

Section 1: What Is AI Prompt Engineering?

Prompt engineering is the practice of writing precise instructions for AI to complete tasks exactly how you want them. Every good output starts with a good prompt. We'll now use prompt engineering to modify UI sections, add features, and deploy improvements.

Tips for Success

- ✅ Be **clear** and **specific** (what do you want and how should it look?)
- ✅ Provide **context** and **intent**
- ✅ **Iterate** to refine results if needed

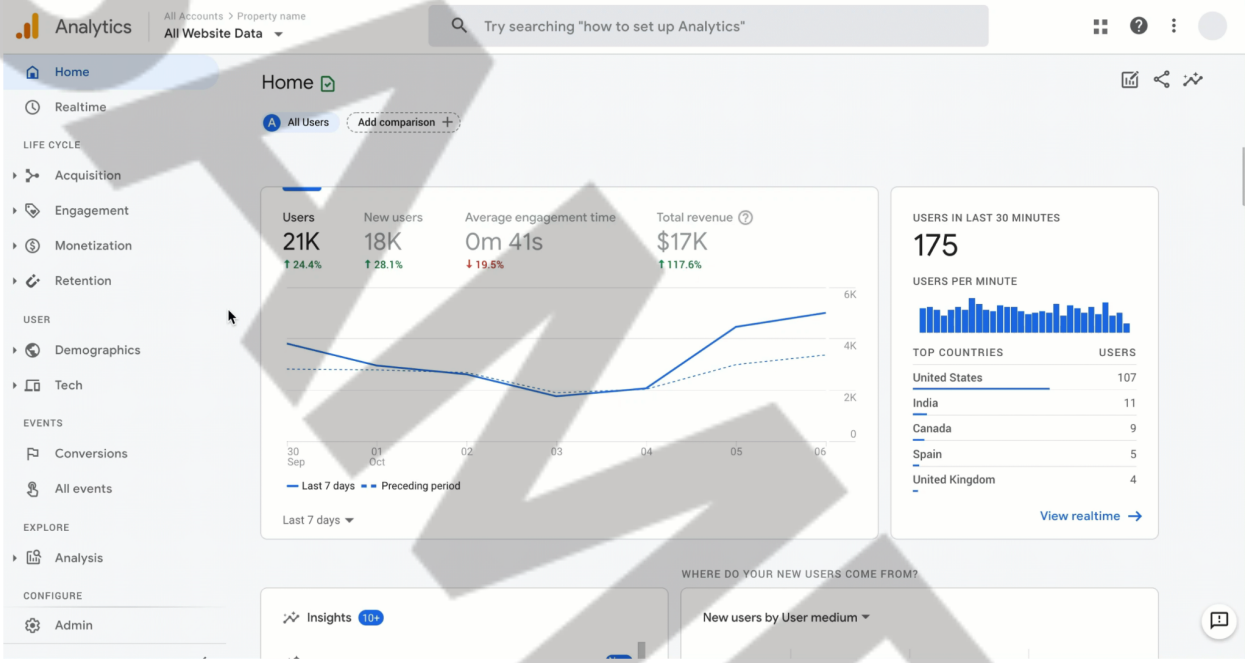
Section 2: Designing the Hero Section

To see full course, buy it at aiwebcourse.com

To see full course, buy it at aiwebcourse.com

THIS IS A VERY IMPORTANT STEP!

You have to set-up Google Analytics and Facebook Pixel if you are planning to purchase ads through Google or Meta. With CLINE this setup is easy and straightforward but this is a necessary step if you want to buy paid traffic later from these platforms!



🧭 Google Analytics Setup Instructions

🔷 Step 1: Create a Google Analytics Account

1. Go to https://analytics.google.com
2. Sign in with your Google account
3. Click **Start Measuring**
4. Enter an account name (e.g., `Ghibli App Analytics`)
5. Choose account data sharing settings and click **Next**

🔷 Step 2: Create a Property

1. Name your property (e.g., `Ghibli Meme App`)
2. Set your **reporting time zone** and **currency**
3. Click **Next** and fill out business information
4. Click **Create**

🔷 Step 3: Get Your GA4 Measurement ID

1. After property creation, click **Web** under “Choose a platform”
2. Enter your website URL (e.g., `https://yourapp.com`)
3. Name your stream (e.g., `Main Web Stream`)
4. Click **Create stream**

➡️ You’ll now see a **Measurement ID** in this format:
`G-XXXXXXXXXX` — copy this value.

Section 7: Creating Visuals with Google Vertex AI

🧪 Creating a Custom Logo with Vertex AI Studio

Step 1: Open Vertex AI Studio

- Go to: https://console.cloud.google.com/
- Enable the **Vertex AI API**
- Open **Vertex AI Studio** from the left sidebar

Step 2: Generate an SVG Logo

Use the **Image Generation** tool, then prompt:

```
Create an SVG logo icon for an app inspired by Studio Ghibli, featuring
whimsical nature elements like trees or a magical creature.
```

💡 Tip: If the tool does not support SVG, generate a PNG and convert it later.

Step 3: Save and Integrate

- Download the logo and save it to: `public/_static/logo.svg`

Prompt CLINE:

```
Update the web app to use the new SVG logo located at /static/logo.svg in
the header and footer.
Ensure it's centered and sized to 40px in height.
```

Section 8: Adding Advanced Features (Stripe + Rate Limiting)

For this section you have to have the STRIPE WEBHOOK LAUNCHED LOCALLY from lesson 2 - Set Up Stripe Webhooks (Local Testing)

This is a difficult section and might be hard to setup, because you need to go through the prompts quite a lot, however you can always refer to aiwebcourse.com video material for specific step-by-step instructions.

🎯 Limit Free Users to 5 Generations

```
Improve rate limiting functionality.
Prevent the user from rendering an image if they've reached their limit,
and return an error before processing starts.
```

💸 Add a Paywall

```
Add a paywall after the free limit is exhausted.
Remove the limit once the user subscribes.
```

🔁 Stripe Webhook and Billing Fixes

```
Update Stripe webhooks to change the user's plan and send a thank-you email
to the user who completed the transaction.
```

```
After the user completes their purchase, redirect them to the dashboard
page.
```

```
Fix the customer portal by configuring the default test mode portal in
Stripe dashboard:
https://dashboard.stripe.com/test/settings/billing/portal
```

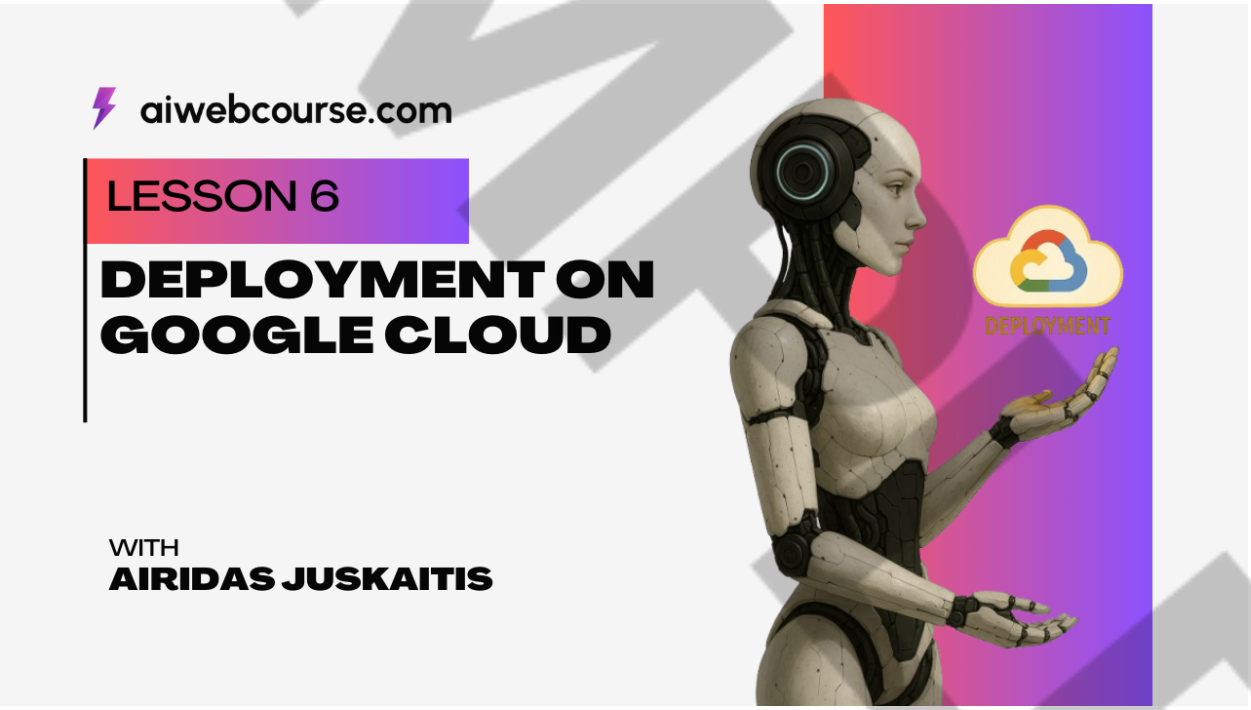
✅ Lesson Recap

By completing Lesson 5, you've:

- Learned prompt engineering through real examples
- Built a visual hero section from AI
- Integrated blog posts and asset folders
- Added tracking with Google Analytics and Facebook Pixel
- Designed your own SVG logo via Vertex AI
- Implemented Stripe paywall logic and rate limits
- Refined header, footer, and overall app UX
- Laid the foundation for a successful launch

Lesson 6: Setting Up the Server and Domain Name

⏱️ Estimated Time: 3–4 hours



🧠 You'll learn how to configure a production server, connect a custom domain, deploy your app from GitHub, and enable SSL using Certbot. We'll also cover how to set up RESEND and Stripe for production.

Section 1: Creating a Virtual Machine (VM) in Google Cloud

To host your application, you need a remote Linux server. We'll use Google Cloud's Compute Engine (VM instance).

🪜 Steps to Create a VM:

1. Go to Google Cloud Console and sign in.
2. In the left sidebar, go to **Compute Engine > VM Instances**.
3. Click **"Create Instance"**.

4. Configure the following settings:

    - **Name**: Pick a name (e.g. `cline-server`).
    - **Region/Zone**: Choose the location closest to your users.
    - **Machine type**: For a small app, `e2-medium` is enough.
    - **Boot disk**: Click **Change** and choose **Debian 12**.
    - **Firewall**: Check both boxes to **allow HTTP and HTTPS traffic**.

5. Scroll down and expand the **Networking** section.
6. Click on the **Network Interface** > **External IP** and select:

    - **Type**: **Static**
    - Click "**Reserve new static IP**" and give it a name (e.g. `cline-static-ip`)
    - This ensures your VM keeps the same IP even after reboot.

7. Click **Create** to launch your virtual machine.

See screenshot below 👇

To see full course, buy it at aiwebcourse.com

To see full course, buy it at aiwebcourse.com

Section 2: Buying a Domain Name (via Namecheap)

To make your app accessible under a real domain like `myapp.com`, you need to connect your domain to the VM IP.

We are using NameCheap because Google Cloud Console does not provide domain registration services.

🪜 Steps:

1. Buy a domain from https://namecheap.com.
2. Go to Namecheap's **Dashboard** > **Domain List**.
3. Click "Manage" for your domain.
4. Scroll to **Advanced DNS**.
5. Add a new A Record:
    - **Type**: A
    - **Host**: `@`
    - **Value**: your Google Cloud VM IP (from previous step)
    - **TTL**: automatic
6. Add a second A Record:
    - **Type**: A
    - **Host**: `www`
    - **Value**: same IP
7. Save changes. DNS propagation may take a few minutes.

    NOTE: You domain name will not become accessible straight away - it takes time for the world wide web to recognize your domain name around the world depending on your location, you can use a tool like https://dnschecker.org/ to track when your domain is available and if it points to the correct IP address (the one you reserved when creating your VM).

Section 3: Preparing the VM Server Using CLINE

At this point, your cloud VM is running — but it's essentially an **empty Linux machine**.

To see full course, buy it at aiwebcourse.com

To see full course, buy it at aiwebcourse.com

To see full course, buy it at aiwebcourse.com

To see full course, buy it at aiwebcourse.com

To see full course, buy it at aiwebcourse.com

🪜 Stripe Setup: Live Payments

Stripe allows you to sell subscriptions to your users. We'll now configure the **Live mode** environment.

🔧 Step 1 – Switch to Live Mode

1. Go to https://dashboard.stripe.com
2. At the top-left corner, **switch to Live Mode**
3. Go to the **Products** tab
4. **Recreate your test plans**:
    - Pro Monthly
    - Pro Yearly
    - Business Monthly
    - Business Yearly

💡 Important: Stripe **test and live modes are completely separate**. You must recreate each plan and get its unique **Price ID**.

📬 Step 2 – Create a Webhook for Your Live App

To let Stripe notify your app of successful payments and subscription updates, set up a **webhook** using your live domain.

1. In the Stripe dashboard, go to **Developers → Webhooks**
2. Click **+ Add endpoint**
3. For the URL, enter:

```
https://yourdomain.com/api/webhooks
```

4. Select the following **event types**:
    - `checkout.session.completed`
    - `invoice.paid`
    - `customer.subscription.updated`
    - `customer.subscription.deleted`

5. Click **Add Endpoint**
6. Copy the **Webhook Signing Secret**, which looks like:

```
whsec_...
```

🗝️ Step 3 – Get Your Production API Keys

Go to **Developers → API Keys** in Live mode:

Copy your **Secret Key**:

```
sk_live_...
```

✅ Step 4 – Update `.env` File with Production Keys

Edit your `.env` again:

```
cd ~/aiwebcourse
```

```
nano .env
```

Update the Stripe section with your live credentials:

```
# -----------
# Subscriptions (Stripe)
# -----------
STRIPE_API_KEY=sk_live_...
STRIPE_WEBHOOK_SECRET=whsec_...

NEXT_PUBLIC_STRIPE_PRO_MONTHLY_PLAN_ID=price_live_...
NEXT_PUBLIC_STRIPE_PRO_YEARLY_PLAN_ID=price_live_...
NEXT_PUBLIC_STRIPE_BUSINESS_MONTHLY_PLAN_ID=price_live_...
NEXT_PUBLIC_STRIPE_BUSINESS_YEARLY_PLAN_ID=price_live_...
```

Replace each placeholder with the actual live values from Stripe.

Save and close the file. *(CTRL + X and Enter)*

🔁 Final Steps – Restart Your App in Production

Now that your `.env` is finalized, the app must be rebuilt and restarted for the changes to take effect.

🛑 1. Stop the Current PM2 Process

If your app is running under PM2, stop it first:

```
pm2 stop aiwebcourse
```

Or if you were running it with `npm start` directly, stop it with `CTRL + C`.

📁 2. Navigate to Your App Folder

```
cd ~/aiwebcourse
```

🔨 3. Rebuild the Application

```
npm run build
```

🚀 4. Start the Application with PM2

```
pm2 start npm --name aiwebcourse -- start
```

💾 5. Save PM2 Process

To ensure your app automatically restarts on server reboot:

```
pm2 save
```

🎉 Congratulations! Your app is now fully:

- Live on a secure domain
- Integrated with **production email**
- Accepting **real payments**
- Automatically managed via **PM2**

You're now running a fully functional SaaS product in production! 🚀

✅ Final Checklist for Production

- Application builds without errors
- Production `.env` is configured
- Domain points to VM IP
- NGINX routes requests properly
- PM2 is running
- SSL certificate is issued
- RESEND DNS is verified
- Stripe is in live mode

🎓 You Made It!

Thank you for learning with me — I truly appreciate the time and effort you've invested in building your AI-powered SaaS application from scratch.

If at any point this PDF guide wasn't detailed enough, don't worry — we've recorded a **full video course** where every step in this guide is replicated **live on screen**. You can access it anytime at - use ***PDFSTUDENT*** discount code to get an even bigger discount for all video and code material:

🌐 Stay Connected

I'd love to see what you've built using this guide — feel free to send me your live apps, progress updates, or even just say hi!

- 🧠 Follow me on X (Twitter): https://x.com/AirisJus
- 💼 Connect with me on LinkedIn: https://www.linkedin.com/in/airidas-juskaitis-a294ab85/
- 📫 Or just drop me an email: **info@aiwebcourse.com**

Let's keep building the future of AI-powered startups together.

See you soon,
– Airidas Jus